WHAT IS CLAIMED IS:

1. A method for solving a computationally intensive problem using a plurality of multipurpose computer workstations, said method comprising the steps of:

(a) building a virtual supercomputer comprising a master computer and at least one slave computer, wherein the at least one slave computer is selected from the plurality of multipurpose computer workstations;

(b) starting a master application on the master computer;

(c) starting a slave application on the at least one slave computer;

(d) dividing the computationally intensive problem into a plurality of task quantum;

(e) assigning to the at least one slave computer at least one task quanta selected from the plurality of task quantum;

(f) completing on the at least one slave computer the at least one task quanta;

(g) receiving on the master computer a result provided by the at least one slave computer; and

(h) repeating steps (e) through (g) until the computationally intensive task is solved.

2. The method of claim 1, wherein the master computer and the at least one slave computer are the same computer.

3. The method of claim 1, further comprising the step of balancing a load on the virtual supercomputer.

4. The method of claim 1, further comprising the steps of:

(a) detecting an event affecting a size of the virtual supercomputer; and

(b) balancing a load on the virtual supercomputer in response to the event.

5. The method of claim 4, wherein the event comprises an availability of a second slave computer selected from the plurality of computer workstations.

6. The method of claim 5, further comprising the step adding the second slave computer to the virtual supercomputer, and wherein the step of balancing a load comprises the steps of:

(a) retrieving the at least one task quanta assigned to the at least one slave computer;

(b) determining a processing capability for the at least one slave computer and the second slave computer; and

(c) distributing task quantum to the at least one slave computer and the second slave computer according to the determined processing capability.

7. The method of claim 4, wherein the event comprises a failure of the slave application on the at least one slave computer.

8. The method of claim 7, further comprising the steps of:

(a) restarting the slave application on the at least one slave computer; and

(b) testing the restarted slave application on the at least one slave computer.

9. The method of claim 4, wherein the event comprises a loss of communications between the master computer and the at least one slave computer.

10. The method of claim 9, further comprising the steps of:

(a) removing the at least one slave computer from the virtual supercomputer;

(b) determining which task quantum selected from the plurality of task quantum have not been completed;

(c) determining a processing capability for any remaining slave computers in the virtual supercomputer; and

(d) distributing at least one uncompleted task quantum to the remaining slave computers according to the determined processing capability.

11. The method of claim 1, further comprising the steps of:

(a) determining a processing capability for the at least one slave computer; and

(b) distributing the plurality of task quantum to the at least one slave computer according to the determined processing capability.

12. The method of claim 1, further comprising the step of collecting on the master computer a plurality of final performance statistics from the at least one slave computer.

13. The method of claim 1, further comprising the step of tearing down the virtual supercomputer.

14. The method of claim 13, wherein the step of tearing down the virtual supercomputer comprises the steps of:

(a) terminating the slave application on the at least one slave computer;

(b) terminating the master application on the master computer; and

(c) terminating the virtual supercomputer daemon application on the at least one slave computer and on the master computer.

15. The method of claim 1, further comprising the step of providing a result to the computationally intensive problem.

16. The method of claim 1, wherein the step of building a virtual supercomputer comprises the steps of:

(a) starting a first virtual supercomputer daemon application on the master computer;

(b) identifying a potential slave computer selected from the plurality of multipurpose workstations;

(c) downloading a second virtual supercomputer daemon application from the master computer to the potential slave computer;

(d) starting the second virtual supercomputer daemon application on the potential slave computer; and

(e) establishing a communications session between the first virtual supercomputer daemon and the second virtual supercomputer daemon.

17. The method of claim 15, further comprising the steps of:

(a) downloading a slave application from the master computer to the potential slave computer;

(b) starting the slave application on the potential slave computer;

(c) testing the slave application on the potential slave computer; and

(d) making the potential slave computer the at least one slave computer in the virtual supercomputer if the testing of the slave application is successful.

18. The method of claim 16, further comprising the step determining a performance capability of the at least one slave computer.

19. The method of claim 16, further comprising the step of repeating steps 16(b) through 16(e) until there are no more potential slave computers available among the plurality of multipurpose workstations.

20. The method of claim 19, further comprising the steps of:

(a) downloading a slave application from the master computer to the potential slave computer;

(b) starting the slave application on the potential slave computer;

(c) testing the slave application on the potential slave computer;

(d) making the potential slave computer the at least one slave computer in the virtual supercomputer if the testing of the slave application is successful;

(e) calculating a performance capability of the at least one slave computer; and

(f) repeating steps (a) through (e) until there are no more potential slave computers available among the plurality of multipurpose workstations.

21. The method of claim 1, wherein the computationally intensive problem comprises optimization of a financial portfolio.

22. The method of claim 1, wherein the computationally intensive problem comprises optimization of supply chain logistics.

23. The method of claim 22, further comprises determining optimal stock levels on a by-item, by-store basis, together with a corresponding trans-shipment plan on a daily basis.

24. The method of claim 1, wherein the computationally intensive problem is used in conjunction with a branch-and-bound approach to optimization.

25. The method of claim 1, wherein the computationally intensive problem is used in connection with large-scale simulations to reduce total run time.

26. The method of claim 1, wherein the computationally intensive problem comprises a network optimization problem.

27. The method of claim 26, wherein the network optimization problem comprises a vehicle routing and scheduling system.

28. The method of claim 1, wherein the computationally intensive problem comprises database searching across a network of workstations.

29. The method of claim 1, wherein the computationally intensive problem comprises database mining across a network of workstations.

30. The method of claim 1, wherein the computationally intensive problem comprises an image analysis program.

31. A method of building a virtual supercomputer comprising a master computer and at least one slave computer selected from a plurality of multipurpose workstations, said method comprising the steps of:

(a) starting a first virtual supercomputer daemon on the master computer;

(b) identifying a potential slave computer selected from the plurality of multipurpose workstations;

(c) starting a second virtual supercomputer daemon on the potential slave computer;

(d) establishing a communications session between the first virtual supercomputer daemon and the second virtual supercomputer daemon;

(e) starting the slave application on the potential slave computer;

(f) testing the slave application on the potential slave computer; and

(g) making the potential slave computer the at least one slave computer in the virtual supercomputer if the testing of the slave application is successful.

32. The method of claim 31, further comprising the step of calculating a performance capability of the at least one slave computer.

33. The method of claim 31, further comprising the step of repeating steps 31(b) through 31(g) until there are no more potential slave computers available among the plurality of multipurpose workstations.

34. The method of claim 31, further comprising the step of downloading a slave virtual supercomputer software from the master computer to the potential slave computer.

35. The method of claim 31, further comprising the step of downloading a slave application from the master computer to the potential slave computer.

36. A method for dynamically reconfiguring a virtual supercomputer while the virtual supercomputer is solving a computationally intensive problem, said method comprising the steps of:

(a) detecting an event affecting a size of the virtual supercomputer;

(b) reassigning a task quantum from a failed slave computer to at least one slave computer in the virtual supercomputer if the event is a slave failure event; and

(c) retrieving a task quantum from at least one slave computer in the virtual supercomputer and assigning the retrieved task quantum to a new slave computer if the event is a newly available slave event.

37. The method of claim 36, further comprising the steps of:

(a) terminating a slave application on the failed slave computer if the event is a slave failure event; and

(b) restarting the slave application on the failed slave computer if the event is a slave failure event,

wherein steps 37(a) and 37(b) are performed before step 36(b) is performed.

38. The method of claim 37, further comprising the steps of:

(a) initializing the restarted slave application if the event is a slave failure event;

(b) testing the restarted slave application if the event is a slave failure event; and

(c) terminating the restarted slave application if the testing step is not successful,

wherein steps 38(a) through 38(c) are performed before step 36(b) is performed.

39. The method of claim 38, further comprising the step of removing the failed slave computer from the virtual supercomputer if the testing step is not successful.

40. The method of claim 36, wherein the step of adding an additional slave computer to the virtual supercomputer if a workstation selected from the plurality of multipurpose workstations becomes newly available comprises the steps of:

(a) starting a virtual supercomputer daemon on the available workstation; and

(b) starting a slave application the newly available workstation.

41. The method of claim 40, further comprising the steps of:

(a) initializing the slave application on the newly available workstation; and

(b) testing the slave application on the newly available workstation.

42. A method for solving a computationally intensive problem using a plurality of multipurpose computer workstations, said method comprising the steps of:

(a) building a virtual supercomputer comprising a master computer and at least one slave computer, wherein the slave computer is selected from the plurality of multipurpose computer workstations;

(b) distributing a plurality of data sets to the at least one slave computer;

(c) balancing an initial load distribution on the virtual supercomputer;

(d) assigning initial computational tasks to the at least one slave computer;

(e) initiating computations on the at least one slave computer;

(f) receiving a first set of results from the at least one slave computer;

(g) adding an additional slave computer to the virtual supercomputer if a workstation selected from the plurality of multipurpose workstations becomes available;

(h) removing a slave computer from the virtual supercomputer if the slave computer becomes unavailable; and

(i) distributing the first set of results and additional data sets to the at least one slave computer until the computationally intensive problem has been solved.

43. The method of claim 42, further comprising the steps of:

(a) collecting a plurality of final performance statistics from each slave computer; and

(b) tearing down the virtual supercomputer.

44. A method for incorporating a new member computer into a virtual supercomputer solving a computationally intensive problem, said method comprising the steps of:

(a) sending a first messages from the new member computer to a master computer in the virtual supercomputer, said first message comprising an availability status for the new member computer;

(b) starting a virtual supercomputer daemon on the new member computer;

(c) starting a slave application on the new member computer;

(d) performing a self-test of the slave application on the new computer;

(e) sending a second message from the new member computer to the master computer, said second message comprising a performance statistic report;

(f) receiving a data set on the new member computer;

(g) receiving a task on the new member computer, said task comprising computational instructions for solving a portion of the computationally intensive problem; and

(h) completing the task on the new member computer.

45. The method of claim 44, further comprising the steps of:

(a) sending a third message from the new member computer to the master computer, said third message comprising a failure report if the self-test is not successful;

(b) terminating the virtual supercomputer daemon on the new member computer if the self-test is not successful; and

(c) terminating the slave application on the new member computer, if the self-test is not successful.

46. The method of claim 44, further comprising the steps of:

(a) collecting a plurality of performance statistics on the new member computer; and

(b) sending a third message from the new member computer to the master computer, said third message comprising the plurality of performance statistics.

47. A method for dynamically adding a new member computer to a virtual supercomputer while the virtual supercomputer is solving a computationally intensive problem, said method comprising the steps of:

(a) detecting the availability of the new member computer;

(b) providing a virtual supercomputer daemon to the new member computer;

(c) instructing the new member computer to start the virtual supercomputer daemon;

(d) providing a slave application to the new member computer;

(e) instructing the new member computer to start the slave application;

(f) redistributing a computational load on the virtual supercomputer; and

(g) performing computational tasks on the virtual supercomputer.

48. The method of claim 47 further comprising the steps of:

(a) providing an initial data set to the new member computer; and

(b) instructing the new member computer to perform a self-test using the initial data set.

49. The method of claim 48 further comprising the steps of:

(a) receiving a self-test result from the new member computer;

(b) instructing the new member computer to terminate the slave application if the self-test is not successful; and

(c) removing the new member computer from the virtual supercomputer if the self-test is not successful.

50. The method of claim 47, wherein the step of redistributing a computational load on the virtual supercomputer comprises the steps of:

(a) retrieving a plurality of task quantum from each member computer in the virtual supercomputer;

(b) determining a computational capability for each member computer in the virtual supercomputer; and

(c) assigning a subset of the plurality of task quantum to each member computer in the virtual supercomputer.

51. A method for dynamically processing a slave application failure on a member computer in a virtual supercomputer while the virtual supercomputer is solving a computationally intensive problem, said method comprising the steps of:

(a) detecting the slave application failure;

(b) instructing the member computer to restart the slave application;

(c) providing an initial data set to the member computer;

(d) instructing the member computer to perform a self-test using the initial data set;

(e) instructing the member computer to continue performing computational tasks for a previously assigned task quanta.

52. The method of claim 51 further comprising the steps of:

(a) receiving a self-test result from the member computer;

(b) instructing the member computer to terminate the slave application if the self-test is not successful;

(c) redistributing a computational load on the virtual supercomputer; and

(d) removing the member computer from the virtual supercomputer if the self-test is not successful.

53. The method of claim 52, wherein the step of redistributing a computational load on the virtual supercomputer comprises the steps of:

(a) determining which of a plurality of task quantum was previously assigned to the member computer;

(b) determining a computational capability for each member computer in the virtual supercomputer; and

(c) assigning a subset of the plurality of task quantum to each member computer in the virtual supercomputer.

54. A method for dynamically processing a termination of a virtual supercomputer daemon on a member computer in a virtual supercomputer while the virtual supercomputer is solving a computationally intensive problem, said method comprising the steps of:

(a) detecting the termination of the virtual supercomputer daemon;

(b) instructing the member computer to restart the virtual supercomputer daemon;

(c) instructing the member computer to restart the slave application;

(d) providing an initial data set to the member computer;

(e) instructing the member computer to perform a self-test using the initial data set;

(f) instructing the member computer to continue performing computational tasks for a previously assigned task quanta.

55. The method of claim 54 further comprising the steps of:

(a) receiving a self-test result from the member computer;

(b) instructing the member computer to terminate the slave application if the self-test is not successful;

(c) redistributing a computational load on the virtual supercomputer; and

(d) removing the member computer from the virtual supercomputer if the self-test is not successful.

56. The method of claim 55, wherein the step of redistributing a computational load on the virtual supercomputer comprises the steps of:

(a) determining which of a plurality of task quantum was previously assigned to the member computer;

(b) determining a computational capability for each member computer in the virtual supercomputer; and

(c) assigning a subset of the plurality of task quantum to each member computer in the virtual supercomputer.

57. A system for solving a computationally intensive problem using a virtual supercomputer, said system comprising:

(a) a virtual supercomputer comprising a master computer in communication with at least one slave computer;

(b) a master application running on the master computer; and

(c) a slave application running on the at least one slave computer;

wherein the master application divides the computationally intensive task into a plurality of task quantum and distributes a subset of the plurality of task quantum to the at least one slave computer, and wherein the slave application performs computations on the subset as directed by the master computer.

58. The system of claim 57, wherein the communication between the master computer and the at least one slave computer comprises a local area network.

59. The system of claim 57, wherein the communication between the master computer and the at least one slave computer comprises a wide area network.

60. The system of claim 57, wherein the communication between the master computer and the at least one slave computer comprises the Internet.

61. The system of claim 57, wherein the master application is adapted to receive periodic messages from the slave application, and wherein in response to the periodic messages, the master application performs a load balancing procedure on the virtual supercomputer.

62. The system of claim 57, wherein the master application detects the availability of a new slave computer and dynamically reconfigures the virtual supercomputer to incorporate the new slave computer in solving the computationally intensive problem.

63. The system of claim 57, wherein the computationally intensive problem comprises optimization of a financial portfolio.

64. The system of claim 57, wherein the computationally intensive problem comprises optimization of supply chain logistics.

65. The system of claim 64, wherein the computationally intensive problem further comprises determination of optimal stock levels on a by-item, by-store basis, together with a corresponding trans-shipment plan on a daily basis.

66. The system of claim 57, wherein the computationally intensive problem is used in conjunction with a branch-and-bound approach to optimization.

67. The system of claim 57, wherein the computationally intensive problem is used in connection with large-scale simulations to reduce total run time.

68. The system of claim 57, wherein the computationally intensive problem comprises network optimization problem.

69. The method of claim 68, wherein the network optimization problem comprises a vehicle routing and scheduling system.

70. The system of claim 57, wherein the computationally intensive problem comprises database searching across a network of workstations.

71. The system of claim 57, wherein the computationally intensive problem comprises database mining across a network of workstations.

72. The system of claim 57, wherein the computationally intensive problem comprises an image analysis program.